Exploring Telecom Customer Churn Prediction with Machine Learning

Md Ishtiaq Ahmed
Computer Science
New Mexico State University
ishtiaq@nmsu.edu

Israel Gonzalez S.
Computer Science
New Mexico State university
igonzals@nmsu.edu

PROBLEM STATEMENT

Predicting Customer Churn[1] in telecom service providers is challenging but essential for improving customer retention, reducing costs, and enhancing financial performance. An accurate predictive model can help companies take preventive measures and retain customers, ultimately increasing profits.

MOTIVATION

Ishtiaq's experience as a data analyst in a Telecom company and Israel's academic experience on Business Intelligence motivated the study, as they recognized the real-world impact of predicting customer churn on business performance and profitability. The problem also offers the opportunity to apply various machine learning techniques.

RELATED WORKS

Numerous research projects have been conducted to predict customer churn in the Telecom industry. One of the research projects we found, titled **"Behavior-Based Telecommunication Churn Prediction with Neural Network Approach" [1],** used neural networks to predict customer churn in a Telecom company. Customer service usage information is utilized as the features. Customer churn was predicted using clustering algorithms.

Another research project was **"Intelligent Decision Forest Models for Customer Churn Prediction" [2]** where the Random Forest, Functional Tree, and Logistic Model tree (LMT) algorithms showed better results than Naïve Bayes (NB) and KNN.

SOLUTION

1.1. Exploratory Data Analysis: This is important to understand the characteristics of data. It helps identify potential issues with the data and can provide insights about the data structure, which is important to select appropriate machine learning models.

[1] Customer who was taking services earlier but not using any service now are considered as churned.

1.2 Cleaning the Dataset: We proceeded to drop unnecessary attributes and samples to efficiently manage the resources to model our problem.

1.3. Sampling and balancing the dataset: We balanced the training data and took a subset of the data for KPCA.

2. Machine Learning Task - Classification: By applying classification techniques, we classified the customers into two classes: *churn* or *not churn.*

Initially, we did some pre-processing on the data set to make it ready to fit in a classification model. At first logistic regression was used. Although this model gave good accuracy, other performance parameters like precision, recall and F1 were not good. Then we balanced the data set and performed dimensionality reduction. Then again, we fit the dimension-reduced data to logistic regression and decision tree classifier. This time all the performance parameters are at par with each other.

DATA DESCRIPTION

The churn of a telecom customer can depend on many things, and it is challenging to predict customer churn. That is why we were looking for a telecom customer data set with many instances along with a good number of features. We found a telecom customer dataset [2] [3] with 99,999 instances and 225 features. Below is a brief description of the dataset:

• Each row of this dataset represents one unique customer.

• All the features are telecom customer attributes related to what services they are using, spending on different services, talk time, data usage, recharge amount, data of last usage, date of recharge, types of data pack, and many others. The data types of the features are:

dtypes: float64(179), int64(35), object(12)

• Every attribute data is for four months.

• One additional column is added to indicate churn. Customers who did not generate any revenue in the month of September fall under churned customers.A detailed data dictionary is provided in a separate Excel file.

[2] A detail data dictionary is provided in a separate excel sheet.

EXPLORATORY DATA ANALYSIS

In the dataset, there were 40 features where more than 70% of the values were *null*. These columns were deleted since they did not add significant value.

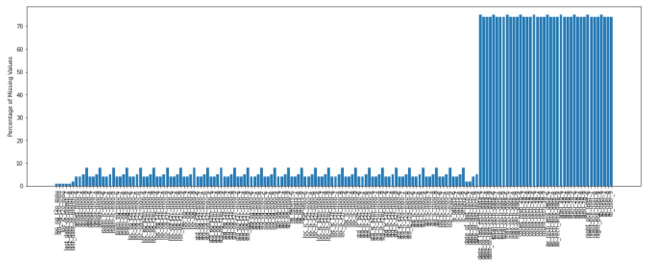

Fig 1: All the features with (%) of Null value

There was a total of 16 features where only a single unique value was *present*. These features were deleted from the dataset. After that 170 features were remaining.

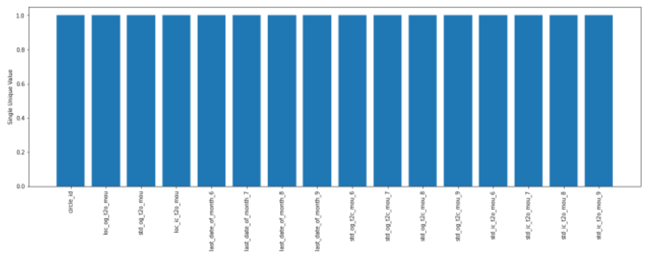

Fig 2: features with unique value

There were many standard ways to fill up the *null* values. We used the mean value of each numeric column to fill up *null* values. After performing the action below, the status of the percentage of *null* value was greater than 0:

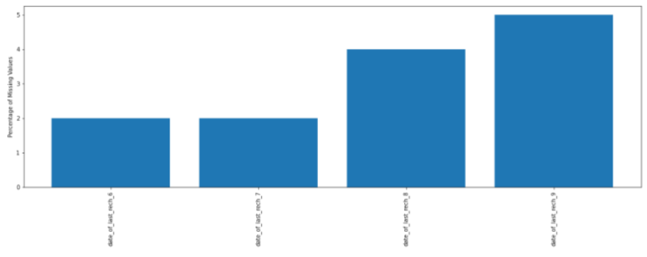

Fig 3: Features with Null value after treatment

So, there were still some *null* values present in the *date* column. Since date values cannot be replaced with mean, we replaced them with zero.

Next, we identified the problem's nature by preparing some scatter plot where we crossed *arpu_8, vol_3g_mb_8, total_og_mou_8*, and *total_rech_amt_8*. Below scatter plot clearly shows that this was a non-linear problem.

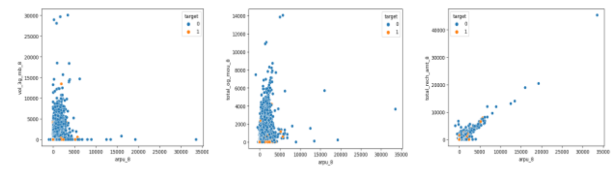

Fig 4: Scatter Plot

RESULT ANALYSIS

Initial Solution: Initially we applied Logistic Regression to see the performance, and 89.8% of accuracy was achieved. Although the accuracy seems good, it is important to look at other performance metrics. Below, are the other metrics:

Classifier: Logistic Regression

Hyper Parameters: C =1 , max_iter = 100

		Result
Accuracy (%)	**Training**	0.898
	Testing	0.898
Precision	**Training**	0.449
	Testing	0.449
Recall	**Training**	0.5
	Testing	0.5
F1	**Training**	0.473
	Testing	0.473
Fitting Time (sec)	**Training**	0.679

Table 1: Performance of Logistic Regression

It was evident from the performance evaluation that we did not achieve the optimal result as precision, recall, and f1 all were low. After further investigation, we found that the prediction column was imbalanced. We have used SMOTE[3] function to balance the data.

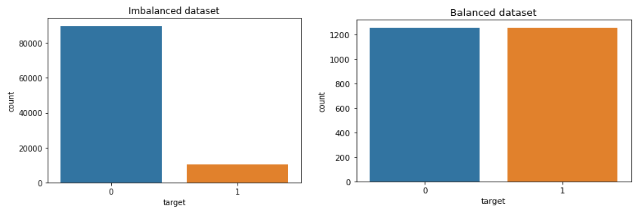

Fig 5: Number of Target before and after data balancing

Dimensionality Reduction: Later, we did Principal Component Analysis to reduce the number of features.

[3] SMOTE is a technique used in machine learning to balance imbalanced datasets by generating synthetic samples for the minority class, improving the performance of classification algorithms.

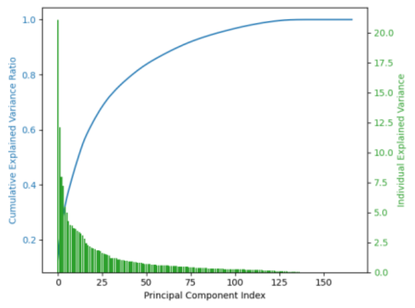

Fig 6: Explained variance vs. Principal component index

From what the chart showed, we could say that using 50 features 85% of the cumulative explained variance ratio can be achieved.

We used **Kernel PCA** for feature reduction. Using PCA earlier, we decided to use *n_components* = 50. We also tune the *kernel* and *gamma* parameters to achieve a better result. We also used *train_test_split* method to get a subset of data to do the KPCA. Otherwise, it could not be possible to run this on a local machine. We selected 2000 instances to do the KPCA.

We fitted the balanced and dimension-reduced data in two classifiers to check the performance.

1. Logistic Regression: For this purpose, we kept the same hyper parameters that were used before data balancing and dimensionality reduction. Below, is the outcome:

Classifier: Logistic Regression

Hyper Parameters: C = 1, max_iter = 100

		Result
Accuracy (%)	**Training**	0.986
	Testing	0.959
Precision	**Training**	0.986
	Testing	0.962
Recall	**Training**	0.986
	Testing	0.959
F1	**Training**	0.986
	Testing	0.959
Fitting Time (sec)	**Training**	0.068

Table 2: Performance of Logistic Regression

We could clearly see that we had achieved better accuracy, precision, recall, and f1 score using the same logistic regression model after data balancing and dimension reduction. The fitting time was also very low. There was a small difference between training and testing accuracy which suggests that the model is not overfitting.

Classifier: Decision Tree

Hyper Parameters: criterion='gini', max_depth=10

		Result
Accuracy (%)	**Training**	0.996
	Testing	0.906
Precision	**Training**	0.996
	Testing	0.907
Recall	**Training**	0.996
	Testing	0.906
F1	**Training**	0.996
	Testing	0.906
Fitting Time (sec)	**Training**	0.174

Table 3: Performance of Decision Tree

The Decision Tree model gave 99.6% accuracy on the training set whereas for the test set the accuracy was 90.6%, which suggests that the model is overfitting. Running time was also very low.

We also used KNN (K-nearest neighbor) classifier to check the performance of the data set:

Classifier: KNN (1)

Hyper Parameters: n_neighbors=5, weights = 'uniform', p=2

		Result
Accuracy (%)	**Training**	0.989
	Testing	0.906
Precision	**Training**	0.989
	Testing	0.906
Recall	**Training**	0.989
	Testing	0.906
F1	**Training**	0.989
	Testing	0.906
Fitting Time (sec)	**Training**	0.002

Table 4: Performance of KNN (1)

Using KNN, the performance was almost identical to the performance of the Decision Tree classifier. After some hyper parameter tuning, we found that with n_neighbors = 40, training and testing accuracy is almost sane.

Classifier: KNN (2)

Hyper Parameters: n_neighbors=40, weights = 'uniform', p=2

		Result
Accuracy (%)	**Training**	0.916
	Testing	0.906
Precision	**Training**	0.922
	Testing	0.907
Recall	**Training**	0.916
	Testing	0.906
F1	**Training**	0.916
	Testing	0.906
Running Time (sec)	**Training**	0.002

Table 5: Performance of KNN (2)

Then, by increasing the *n_neighbors* value, we could reduce overfitting, and thus avoided generalization errors.

Classifier: AdaBoost

Hyper Parameters: base_estimator = base_estimator,n_estimators=50, learning_rate=0.5

We also wanted to test a new classifier to see how it works. And we chose AdaBoost classifier for that. Below is the performance using AdaBoost:

		Result
Accuracy (%)	**Training**	0.983
	Testing	0.954
Precision	**Training**	0.983
	Testing	0.955
Recall	**Training**	0.983
	Testing	0.954
F1	**Training**	0.983
	Testing	0.954
Running Time (sec)	**Training**	0.826

Table 6: Performance of AdaBoost

AdaBoost gives a good test and train accuracy. However, test accuracy is 3% lower than train accuracy which indicates that there is some overfitting but it is better than Decision Tree and KNN(1) where we have observed more than 8% gape between train and test accuracy. On the other hand, fitting time a little higher but still in a considerable range.

CONCLUSIONS

After comparing the outcomes of all the classifiers, it seems that Logistic Regression gives a better result. However, Logistic Regression on a non-linear dataset depends on the specific problem and dataset. While it may perform well in some cases, it might not always be the best choice for non-linear problems as it is mainly to solve linear classification problems and we used this classifier to experiment. Considering this, we conclude that KNN with *n_neighbors* = 40 is a good choice to solve this problem. However, We also achieved good results using the AdaBoost classifier. Accuracy improved significantly however it may create some generalization errors. To conclude we can use KNN (2) or AdaBoost for this problem depending on the requirements.

REFERENCES

[1] Yongbin Zhang; Ronghua Liang; Yeli Li; Yanying Zheng and Michael Berry 20011. Behavior-Based Telecommunication Churn Prediction with Neural Network Approach. IEEE Xplore: 25 August 2011. DOI: https://doi.org/10.1109/ISCCS.2011.89

[2] Fatima Enehezei Usman-Hamza, Abdullateef Oluwagbemiga Balogun, Luiz Fernando Capretz, Hammed Adeleye Mojeed 2022. Intelligent Decision Forest Models for Customer Churn Prediction. Appl. Sci. 2022, 12(16), 8270. DOI: https://doi.org/10.3390/app12168270

[3] Link of the data set: https://data.world/kishoresjv/telecomchurn/workspace/file?filename=telecom_churn_data.csv